Data Services Leveraging Bing's Data Assets

Kaushik Chakrabarti, Surajit Chaudhuri, Zhimin Chen, Kris Ganjam, Yeye He

Microsoft Research

Redmond, WA

{kaushik, surajitc, zmchen, krisgan, yeyehe}@microsoft.com

Abstract

Web search engines like Bing and Google have amassed a tremendous amount of data assets. These include query-click logs, web crawl corpus, an entity knowledge graph and geographic/maps data. In the Data Management, Exploration and Mining (DMX) group at Microsoft Research, we investigate ways to mine the above data assets to derive new data that can provide new value to a wide variety of applications. We expose the new data as cloud data services that can be consumed by Microsoft products and services as well as third party applications. We describe two such data services we have built over the past few years: synonym service and web table service. These two data services have shipped in several Microsoft products and services including Bing, Office 365, Cortana, Bing synonyms API and Bing Knowledge API.

1 Introduction

Web search engines like Bing and Google have amassed a "treasure trove" of data assets. One of the most important assets is the query-click log which contains every search query submitted by a user, the urls and other information (e.g., answers) returned by the search engine, and the items clicked on by the user. Other important assets include the web crawl corpus, an entity knowledge graph (that contains information about named entities like people, places and products) and geographic/maps data.

The above data assets are leveraged by the web search engine to deliver a high quality search experience. For example, query-click log is used to improve the quality of web result ranking. The entity knowledge graph is used not only to improve web result ranking but also to compose the "entity information card" for entity queries. In the Data Management, Exploration and Mining (DMX) group at Microsoft Research, we explore ways to mine the above data assets to *derive new data* that can provide new value to a wide variety of applications. We expose the new data as cloud data services that can be consumed by Microsoft products and services as well as third party products and services. The main idea is depicted in Figure 1.

Synonym service: Let us start with an example of such a data service called synonym service. People often refer to a named entity like a product or a person or a place in many different ways. For example, the camera 'Canon 600d' is also referred to as 'canon rebel t3i', the film 'Indiana Jones and the Kingdom of the Crystal Skull' also as 'indiana jones 4' the person 'Jennifer Lopez' also as 'jlo' and the place 'Seattle Tacoma International Airport' also as 'sea tac'. We refer to them as entity synonyms or simply synonyms (in contrast to other types

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

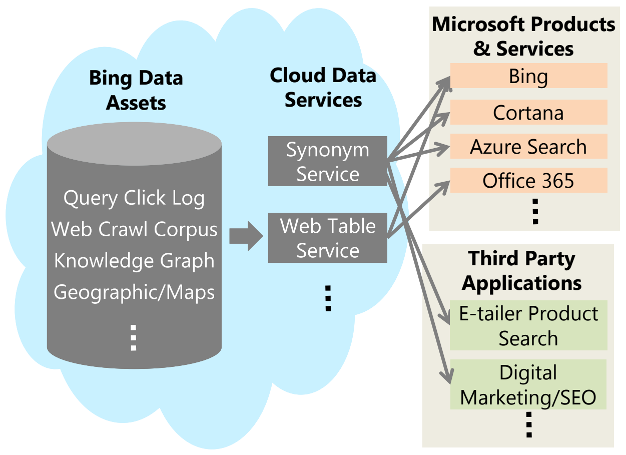


Figure 1: Data services leveraging Bing's data assets

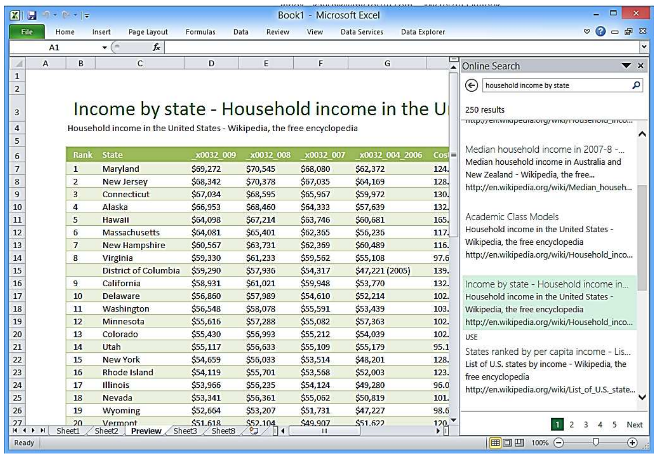
of synonyms such as attribute synonyms [15]). Consider the product search functionality on an e-tailer site like bestbuy.com. Without the knowledge of synonyms, it often fails to return relevant results. For example, when the user searches for 'indiana jones and kingdom of crystal skull' on bestbuy.com, it returns the DVD for that movie (the desired result). However, if she chooses to search using the query 'indiana jones 4', it fails to return the desired result. This is because bestbuy.com does not have the knowledge that the above film is also referred to as 'indiana jones 4'. If we can create a data service that computes the synonyms for any entity by leveraging the rich data assets of a search engine, it will bring tremendous benefit to such e-tailers [9]. It will also be valuable to specialized entity portals/apps like movie portals (Fandango, Moviefone), music portals (Pandora, Spotify), sports portals (ESPN, NFL) and local portals (Yelp, Tripadvisor). Hence, we built such a data service called the synonym service [7, 18].

Challenges in synonym service: The main challenge is to mine synonyms in a domain independent way with high precision and good recall. While there is significant work on finding similar/related queries [3, 4, 12], there is little work on mining synonyms with the above precise definition of synonymity (i.e., alternate ways of referring to the *exact same entity*). Furthermore, existing synonym mining works rely mostly on query co-clicks [10], which we find in practice to be insufficient to ensure very high precision (e.g., over 95%) that is required for scenarios described above. In our work, we develop a variety of novel features to complement query log features that achieve high precision and recall. We describe those techniques in detail in Section 2.

Impact of synonym service: The synonym service is being used extensively by Microsoft products and services as well as by external customers. Many Bing verticals like sports and movies use the synonym data to improve their respective vertical search qualities. In Bing Sports for example, when a user asks the query 'tampabaybucs', entity synonyms will help to trigger the entity card for "Tampa Bay Buccaneers". Our synonyms are also used inside Bing's entity-linking technology which in turn is used in several applications like Bing Snapp [1], Ask Cortana [2], and Bing Knowledge API [3]. For external customers (e.g., e-tailers), synonym technologies can be accessed from the Bing developer center as a web service called Bing synonym API [1]. An entity name can be submitted as input, and all synonyms of that entity will be returned by the service. This is used in customer scenarios such as query enrichment and catalog enrichment. Thousands of customers subscribe to this API.

Web table services: A second data service [4] we have built is the web table service. Although the web crawl corpus mostly consists of unstructured data like textual documents, images and videos, there is also a vast amount of structured data embedded inside the html documents. For example, there are more than a billion html tables,

[1]https://www.microsoft.com/en-us/store/p/snapp/9nblggh09m4d
[2]https://www.microsoft.com/en-us/mobile/experiences/cortana/
[3]https://www.bing.com/widget/knowledge
[4]Actually two closely related services



Income by state - Household income in the U

Household income in the United States - Wikipedia, the free encyclopedia

Rank	State	_x0032_009	_x0032_008	_x0032_007	_x0032_004_2006	Cos
1	Maryland	$69,272	$70,545	$68,080	$62,372	124.
2	New Jersey	$68,342	$70,378	$67,035	$64,169	128.
3	Connecticut	$67,034	$68,595	$65,967	$59,972	130.
4	Alaska	$66,953	$68,460	$64,333	$57,639	132.
5	Hawaii	$64,098	$67,214	$63,746	$60,681	165.
6	Massachusetts	$64,081	$65,401	$62,365	$56,236	117.
7	New Hampshire	$60,567	$63,731	$62,369	$60,489	116.
8	Virginia	$59,330	$61,233	$59,562	$55,108	97.6
	District of Columbia	$59,290	$57,936	$54,317	$47,221 (2005)	139.
9	California	$58,931	$61,021	$59,948	$53,770	132.
10	Delaware	$56,860	$57,989	$54,610	$52,214	102.
11	Washington	$56,548	$58,078	$55,591	$53,439	103.
12	Minnesota	$55,616	$57,288	$55,082	$57,363	102.
13	Colorado	$55,430	$56,993	$55,212	$54,039	102.
14	Utah	$55,117	$56,633	$55,109	$55,179	95.1
15	New York	$54,659	$56,033	$53,514	$48,201	128.
16	Rhode Island	$54,119	$55,701	$53,568	$52,003	123.
17	Illinois	$53,966	$56,235	$54,124	$49,280	96.0
18	Nevada	$53,341	$56,361	$55,062	$50,819	101.
19	Wyoming	$52,664	$53,207	$51,731	$47,227	98.6
20	Vermont	$51,618	$52,104	$49,907	$51,622	120.

household income by state

250 results

Median household income in 2007-8 -...
Median household income in Australia and New Zealand - Wikipedia, the free...
http://en.wikipedia.org/wiki/Median_househ...

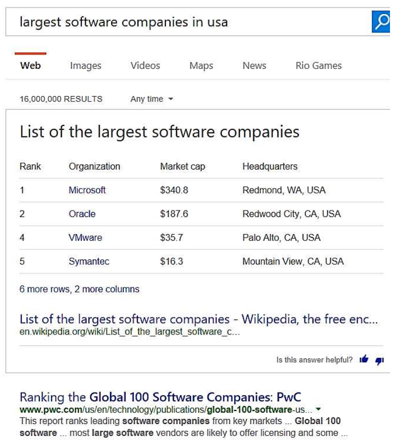
Academic Class Models
Household income in the United States - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Household_inco...

Income by state - Household income in...
Household income in the United States - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Household_inco...
USE

States ranked by per capita income - Lis...
List of U.S. states by income - Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/List_of_U.S._state...

1 2 3 4 5 Next

(a)



largest software companies in usa

16,000,000 RESULTS

List of the largest software companies

Rank	Organization	Market cap	Headquarters
1	Microsoft	$340.8	Redmond, WA, USA
2	Oracle	$187.6	Redwood City, CA, USA
4	VMware	$35.7	Palo Alto, CA, USA
5	Symantec	$16.3	Mountain View, CA, USA

6 more rows, 2 more columns

List of the largest software companies - Wikipedia, the free enc...
en.wikipedia.org/wiki/List_of_the_largest_software_c...

Ranking the **Global 100 Software Companies**: PwC
www.pwc.com/us/en/technology/publications/**global-100-software**-us...
This report ranks leading **software companies** from key markets ... **Global 100 software** ... most **large software** vendors are likely to offer licensing and some ...

(b)

Figure 2: (a) Web table search service in action in Excel PowerQuery (b) Web table answer service in action on Bing.

html lists and spreadsheets on the web. Each such table/list/spreadsheet contains a set/list of named entities and various attributes about those entities. For example, the html table embedded inside `en.wikipedia.org/wiki/List_of_U.S._states_by_income` contains the median household income for all the U.S. states. These tables are often very valuable to information workers. For example, a business data analyst often needs to "join" business data with public data. Consider a data analyst analyzing sales numbers from various U.S. states (say, in Excel) and wants to find how strongly they are correlated with the median household income. She needs to join the former with the table in `en.wikipedia.org/wiki/List_of_U.S._states_by_income`. It is hard for the analyst to discover the above table and also to import it into Excel in order to do the join. It would be very valuable if we can index such tables and allow Excel users to easily search for them (e.g., using keyword search) as shown in Figure 2(a). We built a data service called *web table search service* for the above purpose.

A substantial fraction of queries on Bing and Google can be best answered using a table. For example, for the query 'largest software companies in usa', it is much better to show the table from Wikipedia containing the largest software companies (`en.wikipedia.org/wiki/List_of_the_largest_software_companies`) than just the blue links as shown in Figure 2(b). We refer to the above class of queries as *list-intent queries* as the user is looking for a *list* of entities. We built a data service called *web table answer service* for the above purpose. We explore other classes of queries as well for table answers. Although both web table search and web table answer services take a keyword query as input and returns web tables, they are quite different. The former always returns a *ranked list of tables* relevant to the query. On the other hand, since the latter is invoked by a web search engine where the top result spot is reserved for the best possible answer (among all possible types of answers as well as top algorithmic search result), the desired output for the latter is a *single table* if is the best possible answer, otherwise it should return nothing.

Challenges in web table services: Most of raw HTML tables (i.e., elements enclosed by the <table></table> tags) do not contain valuable data but are used for layout purposes. We need to identify and discard those tables. Furthermore, among the valuable tables, there are multiple different types. We need to distinguish among them in order to understand their semantics which in turn is necessary to provide high quality table search and table answer services. These are challenging problems as they cannot be accomplished via simple rules [5]. Furthermore, providing a high quality table ranking as well as providing table answers with high precision and good coverage are hard problems as well. While there is extensive prior work on table extraction [6, 5, 21, 20, 14, 11], there is limited work on the latter two challenges: table ranking and providing table answers with high precision. In Section 3, we present our table extraction techniques and highlight their differences

with prior table extraction work. We also present the novel approaches we have developed for the latter two challenges.
Impact of web table services: The web table search service was released in Excel PowerQuery in 2013 [2]. It allows Excel users to search and consume public tables directly from Excel. A screenshot is shown in Figure 2(a). The web table answer service has been shipping in Bing since early 2015. It shows table answers for list-intent and other types of queries with 98% precision and with a current coverage of 2%. A screenshot is shown in Figure 2(b).

2 Synonym Service

Given an entity name, the synonym service returns all the synonyms of the entity. We mine all possible synonym pairs in an offline process (200 million pairs in our latest version); the service simply performs a lookup into that data. Currently the service is hosted as a public Bing service in Bing Developer Center [1]. We focus on the key technologies used in offline mining process in the rest of this section.

2.1 Synonym Mining Requirements

Based on the intended use cases, we summarize key requirements of synonym mining as follows.

Domain independence. Entity synonyms are ubiquitous in almost all entity domains. A natural approach is to leverage authoritative data sources specific to each entity domain to generate synonyms. For example, one may use extraction patterns specific to IMDB for movie synonyms. However, techniques so developed are specific to one domain that cannot easily generalize to different domains. Given the scale and the variety of the synonyms we are interested in, developing and maintaining specific techniques for each domain is unlikely to scale. Our goal is to develop domain-independent methods to systematically harvest synonyms for all domains.

High precision. Since the output of our service is used by an array of Microsoft products and third party retailers, who would for example use synonyms to enrich their product catalogs, the precision of our synonyms needs to be very high (e.g., above 95%). Entities that are only related but not equivalent (e.g., "Microsoft office 2015" and "Microsoft office 2013") should not be considered as synonyms, for otherwise they will adversely affect downstream applications like product search.

Good recall. In addition to high precision, we want to discover as many synonyms as possible. The types of synonyms we are interested in ranges from simple syntactic name variations and misspellings (e.g., "Cannon 600d" for the entity "Canon 600d"), to subset/superset variations (e.g., "Canon EOS 600d" for the entity "Canon 600d"), to more semantic synonyms (e.g., "Canon rebel t3i" or "t3i slr" for the entity "Canon 600d"). The synonyms we produce should ideally cover all these types.

Freshness. Since new entities (movies, products, etc.) are constantly being created, and new names coined for existing entities, we want the synonym data to be up-to-date. The mining process thus needs to be refreshed regularly to reflect recent updates, and hence needs to be easily maintainable with minimal human intervention.

2.2 Prior Work on Synonym Mining

Prior work on discovering entity synonyms relies on query co-clicks [10]. Our experience suggests that this alone often leads to many false positives. For example, name pairs like "iphone 6" and "iphone 6s", or "Microsoft office 2015" and "Microsoft office 2013" share significant co-clicks and are almost always predicted as synonyms. We find synonyms so generated to have considerably lower precision than the 95% requirement, and incorrect synonyms like the ones above are particularly damaging to application scenarios such as product catalog enrichment. In this work we develop novel features utilizing a variety of orthogonal data assets to overcome the limitations of query logs.

The problem of finding semantically similar/related queries is related to synonym-finding, and is extensively studied in the literature [3, 4, 12]. The fuzzy notion of semantic relatedness used in this body of work, however,

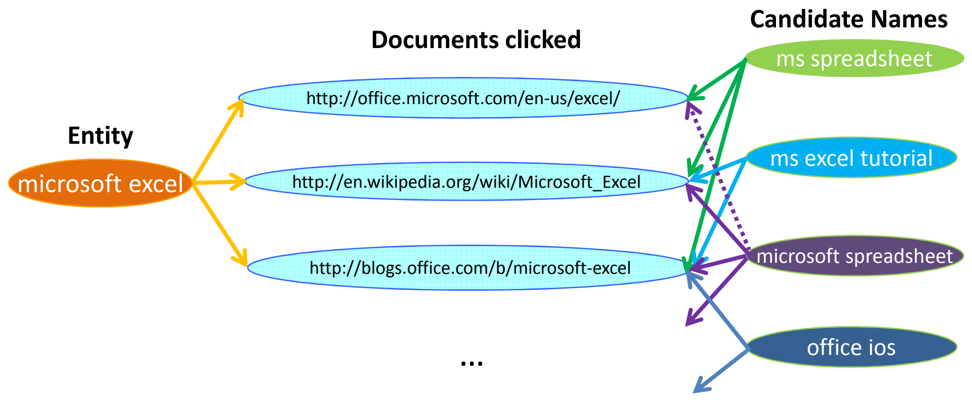

Candidate Names
Documents clicked
ms spreadsheet
http://office.microsoft.com/en-us/excel/
Entity
ms excel tutorial
microsoft excel
http://en.wikipedia.org/wiki/Microsoft_Excel
http://blogs.office.com/b/microsoft-excel
microsoft spreadsheet
office ios
...

Figure 3: Example query log click graphs

does not match our precise requirement of entity-equivalence for synonyms, and is thus insufficient for high precision entity synonyms that we intend to produce.

2.3 Exploiting Bing Data Assets for Synonym Mining

At a high level, our synonym mining has three main steps: (1) generate billions of candidate name pairs that may be synonyms; (2) for each candidate pair, compute rich features derived from data assets such as Bing query logs, web tables, and web documents; (3) utilize manually labeled training data and machine learning classifiers to make synonym predictions for each candidate pair.

We start by generating pairs of candidate names for synonyms. In order to be inclusive and not to miss potential synonyms in this step, we include all pairs of queries from the query logs that clicked on the same document for at least 2 times. This produces around 3 billion candidate synonym pairs.

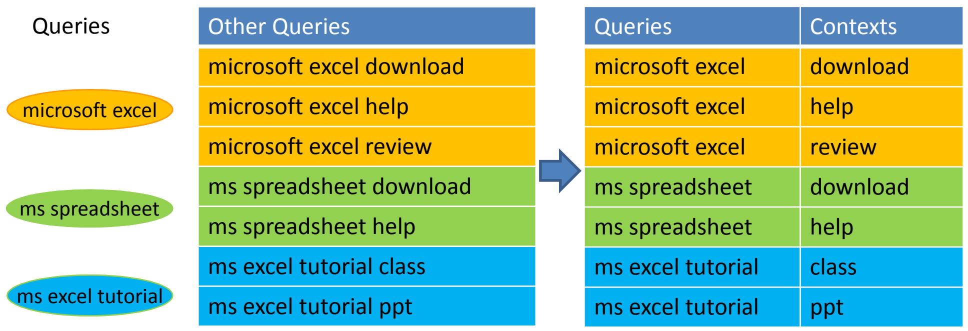
For each candidate pair, we then compute a rich set of features derived from various data sources. Given the feature vectors, we use training data and boosted regression tree [13] to train a binary classifier to predict synonyms. Since boosted regression tree is a standard machine learning model, we will focus our discussions on the design of features using various data sets.

Query log based features. Query logs are one of the most important data assets for synonym mining. The key idea here is the so-called "query co-clicks" [7, 10]. Namely, if search engine users frequently click on the same set of documents for both query-A and query-B, then these two query strings are likely to be synonyms. The rationale here is that search engines clicks form implicit feedback of query-document relevance, which when aggregated over millions of users and a long period of time, provide robust statistical evidence of synonymity between two query strings.

In the example of Figure 3, suppose "microsoft excel" is the entity of interest. For this query users click on three documents in the middle as represented by their urls. If we look at other queries whose clicks share at least one document, we can see that "ms spreadsheet" clicks on the exact same set of documents (a true synonym of "microsoft excel"). Both "microsoft spreadsheet" and "ms excel tutorial" share two co-clicks with "microsoft excel". While the first query is a true synonym, the second is only a related entity (tutorial) and thus not a synonym. Lastly, query "office ios" shares only one clicked document with "microsoft excel", indicating a lower degree of semantic relationship.

Intuitively, the higher the overlap between the clicks shared by two query strings, the more likely they are actual synonyms. We use a number of metrics to quantify relationships between two queries – Jaccard similarity and Jaccard containment when representing their clicked documents as sets, and Jenson-Shannon divergence when representing click frequencies on documents as distributions.

We further encode, for each query, frequency-based features such as the number of clicks, the number of



Other Queries
microsoft excel download
microsoft excel help
microsoft excel review
ms spreadsheet download
ms spreadsheet help
ms excel tutorial class
ms excel tutorial ppt

Queries	Contexts
microsoft excel	download
microsoft excel	help
microsoft excel	review
ms spreadsheet	download
ms spreadsheet	help
ms excel tutorial	class
ms excel tutorial	ppt

Figure 4: Example query contexts

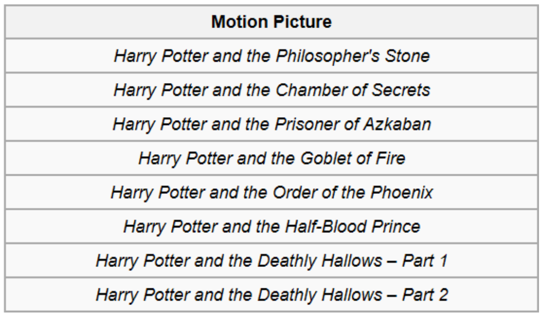
distinct documents clicked and domains clicked, etc. as additional features, which allow machine learning models to differentiate popular queries from less popular ones for example.

One problem we encounter in using query logs is that click-edges in the query-document bipartite graph are sometimes too sparse. To overcome sparsity, we enhance the co-click graph by introducing artificial edges using *pseudo-documents* [7]. Specifically, we construct a pseudo-document $p(d)$ for each clicked document d as the union of the tokens in queries that click on d. Based on this definition, in Figure 3 for example, let d_1 be the first document, then $p(d_1) = \{\text{microsoft, excel, ms, spreadsheet}\}$. We can then add an artificial edge between query q and document d if the tokens of q are contained by the pseudo-document $p(d)$. In Figure 3, because the tokens of query "microsoft spreadsheet" is contained by $p(d_1)$, we add an edge between the two (shown by dotted edge) as if there exists such a click in the original click graph. Using this enhanced bipartite graph, we can define set-based similarity and distribution-based similarity like before. These form a group of pseudo-document-based features.

Notice that in the example of Figure 3, "ms excel tutorial" and "microsoft excel" are related entities with relatively high co-clicks, which however should not be recognized as synonyms because they are of different entity-types (e.g., one is software whereas the other is a tutorial). Similarity based features alone may mistakenly predict them as synonyms. In order to penalize pairs of candidates with different types, we introduce query-context based features. Figure 4 gives an example of query-contexts. For each target query (e.g., "microsoft excel", "ms spreadsheet", etc.), we find from the query logs additional queries that have the target query as a prefix or suffix. For "microsoft excel", we find queries like "microsoft excel download" and "microsoft excel help". From these, we compute the suffix-context of query "microsoft excel" as "download", "help" and "review" (and their respective frequencies). These are indicative of query types, because queries of the same entity type tend to share similar contexts. In this example the context of "microsoft excel" is very similar to that of "microsoft spreadsheet", showing that they are likely of the same entity type. However the context of "microsoft excel" is quite different from "ms excel tutorial", which we use as evidence that the pair may be of different types. We compute distributional similarity between two candidates' query contexts such as Jensen-Shannon divergence [16] as features for type similarity.

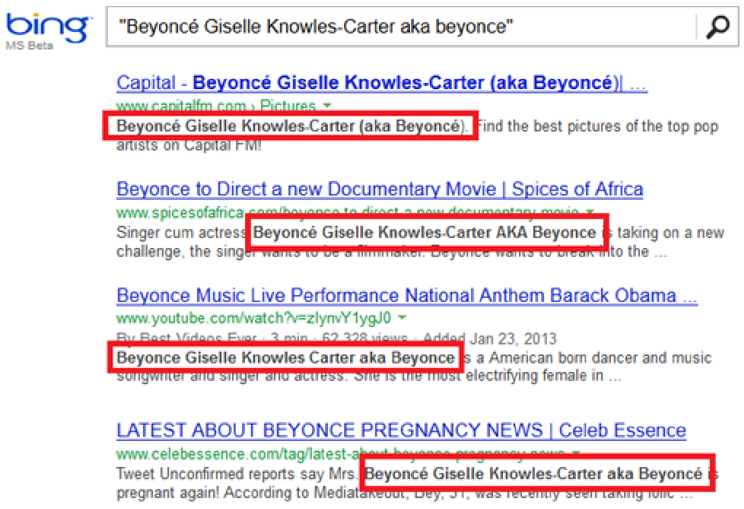
Web table based features. While query logs based features are powerful positive signals for semantic similarity, in many cases they are insufficient to differentiate between true synonyms, and pairs of highly related names that are non-synonyms. For example, the pair "Harry Potter and the Deathly Hallows: Part 1" and "Harry Potter and the Deathly Hallows: Part 2" share substantial co-clicks in the query logs. It is thus difficult to know from the query logs alone that they should not be synonyms.

For this we leverage another set of signals from HTML tables extracted from web pages indexed by Bing. Specifically, we observe that if two entities occur more frequently (than pure coincidence) in the same table columns, they are likely to be different entities and thus not synonyms. The reasoning is that humans are unlikely to put two synonymous mentions of the same entity in the same table column. In a real web table example shown in Figure 5(a), the fact that "Harry Potter and the Deathly Hallows: Part 1" and "Harry Potter



Motion Picture
Harry Potter and the Philosopher's Stone
Harry Potter and the Chamber of Secrets
Harry Potter and the Prisoner of Azkaban
Harry Potter and the Goblet of Fire
Harry Potter and the Order of the Phoenix
Harry Potter and the Half-Blood Prince
Harry Potter and the Deathly Hallows – Part 1
Harry Potter and the Deathly Hallows – Part 2

(a) Value co-occurrence in tables as negative features



"Beyoncé Giselle Knowles-Carter aka beyonce"

Capital - **Beyoncé Giselle Knowles-Carter (aka Beyoncé)**| ...
www.capitalfm.com › Pictures
Beyoncé Giselle Knowles-Carter (aka Beyoncé). Find the best pictures of the top pop artists on Capital FM!

Beyonce to Direct a new Documentary Movie | Spices of Africa
www.spicesofafrica.com/beyonce-to-direct-a-new-documentary-movie
Singer cum actress Beyoncé Giselle Knowles-Carter AKA Beyonce is taking on a new challenge, the singer wants to be a filmmaker. Beyonce wants to break into the ...

Beyonce Music Live Performance National Anthem Barack Obama ...
www.youtube.com/watch?v=zlynvY1ygJ0
By Best Videos Ever · 3 min · 62,328 views · Added Jan 23, 2013
Beyonce Giselle Knowles Carter aka Beyonce is a American born dancer and music songwriter and singer and actress. She is the most electrifying female in ...

LATEST ABOUT BEYONCE PREGNANCY NEWS | Celeb Essence
www.celebessence.com/tag/latest-about-beyonce-pregnancy-news
Tweet Unconfirmed reports say Mrs. Beyoncé Giselle Knowles-Carter aka Beyoncé is pregnant again! According to Mediatakeout, Bey, 31, was recently seen taking folic ...

(b) AKA text patterns

Figure 5: Web tables and document features

and the Deathly Hallows: Part 2" are co-occurring in the same table column indicates that they are likely to be distinct entities instead of synonyms.

We aggregate such statistical information from over 100 million tables extracted from the web, and use the point-wise mutual information of two candidate names occurring in the same table columns as an additional feature.

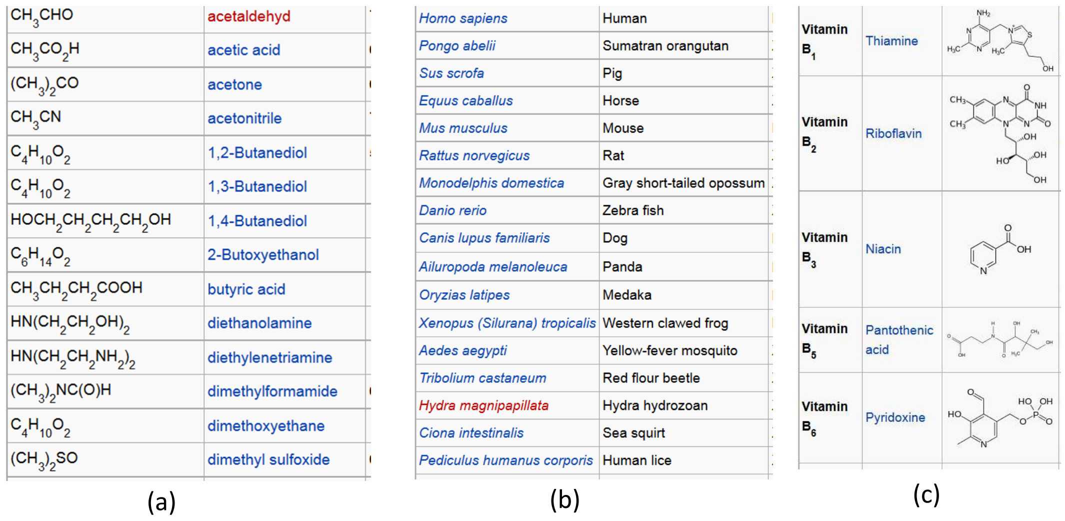
Another way in which we can leverage web tables is to directly utilize synonymous column pairs often seen in tables. Figure 6 gives a few example tables in different domains where two values from the same row are synonyms. We use an initial version of synonym scores to discover such pairs of synonymous columns from the web tables corpus, and then compute the number of times name-A and name-B occur in same rows of synonymous columns as additional features. We find this feature to be helpful in improving coverage of tail entities.

Web document based features. Web documents indexed by Bing provide an orthogonal source of signals. In particular, we consider two main groups of signals: text patterns, and anchor texts.

For text patterns, we scan all documents in Bing's crawl for patterns such as "name-A, also known as name-B", "name-A, aka name-B", "name-A, otherwise known as name-B", etc. Given the huge variety and extensive coverage of the Web documents, this helps us to capture a wide range of synonymous names. Figure 5b shows an example to illustrate this idea. The names "beyonce giselle knowles carter" and "beyonce" frequently co-occur in these AKA text patterns, and they are indeed synonyms. We simply count the number of times name-A and name-B occur in such patterns in all Bing documents as a feature for these two names. We note that the idea of using text patterns is well-studied especially in the literature of information extraction (e.g., [8, 17]). We leverage such patterns in the context of synonym mining for complementary signals derived from an orthogonal data source to achieve improved result quality.

For anchor texts, we utilize the observation that anchor texts describing the same hyper-links are often interchangeable in meanings. So for each hyperlink in Wikipedia, we extract all anchor texts associated with that link, and count the number of times name-A and name-B are used as anchor texts pointing to the same link. For instance, both "Jennifer Lopez" and "JLO" are used as anchor texts describing the link pointing to her Wikipedia page, and we use their respective frequencies pointing to the same page as features. This is another beneficial feature derived from distant supervision.

Wikipedia and Wiktionary. Wikipedia has rich redirect information. For example, "J.Lo" and "JLO" are redirected to the "Jennifer Lopez" page on Wikipedia. We extract such information from a dump of the



CH_3CHO	acetaldehyd
CH_3CO_2H	acetic acid
$(CH_3)_2CO$	acetone
CH_3CN	acetonitrile
$C_4H_{10}O_2$	1,2-Butanediol
$C_4H_{10}O_2$	1,3-Butanediol
$HOCH_2CH_2CH_2CH_2OH$	1,4-Butanediol
$C_6H_{14}O_2$	2-Butoxyethanol
$CH_3CH_2CH_2COOH$	butyric acid
$HN(CH_2CH_2OH)_2$	diethanolamine
$HN(CH_2CH_2NH_2)_2$	diethylenetriamine
$(CH_3)_2NC(O)H$	dimethylformamide
$C_4H_{10}O_2$	dimethoxyethane
$(CH_3)_2SO$	dimethyl sulfoxide

(a)

Homo sapiens	Human
Pongo abelii	Sumatran orangutan
Sus scrofa	Pig
Equus caballus	Horse
Mus musculus	Mouse
Rattus norvegicus	Rat
Monodelphis domestica	Gray short-tailed opossum
Danio rerio	Zebra fish
Canis lupus familiaris	Dog
Ailuropoda melanoleuca	Panda
Oryzias latipes	Medaka
Xenopus (Silurana) tropicalis	Western clawed frog
Aedes aegypti	Yellow-fever mosquito
Tribolium castaneum	Red flour beetle
Hydra magnipapillata	Hydra hydrozoan
Ciona intestinalis	Sea squirt
Pediculus humanus corporis	Human lice

(b)

Vitamin B_1	Thiamine	
Vitamin B_2	Riboflavin	
Vitamin B_3	Niacin	
Vitamin B_5	Pantothenic acid	
Vitamin B_6	Pyridoxine	

(c)

Figure 6: Example tables with synonymous columns

Wikipedia pages as features. Synonyms as defined by traditional thesaurus are also useful. We parse such information from Wiktionary as additional features.

Other syntactic features. As additional negative signals, we describe the syntactic difference between two names using character types and lengths, to leverage observations such as if two names only differ by a number, they are unlikely to be synonyms (e.g., “ford mustang” and “ford mustang 2015”).

Quality evaluation. After producing feature vectors for all pairs of candidate names, we train a boosted regression tree model [13] to predict synonymity for each name pair. Based on our calibration of classification scores using a holdout set of labeled data, we threshold to produce over 200 million synonym pairs with very high precision.

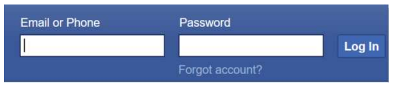
We take a sample from the produced name pairs and perform manual labeling to assess result quality. The result is shown in Table 1. Recall here is defined as simply the number of synonyms produced for each sampled name, where tier-1 recall only counts miss-spellings and name variations (“Canon 600d” and “Cannon 600d”), tier-2 only counts subset/superset synonyms (“Canon 600d” and “Canon EOS 600d”), and tier-3 counts semantic synonyms that do not belong to the two previous categories (“Canon 600d” and “EOS rebel t3i”).

Precision	Tier-1 recall	Tier-2 recall	Tier-3 recall
0.973	1.43	2.12	1.12

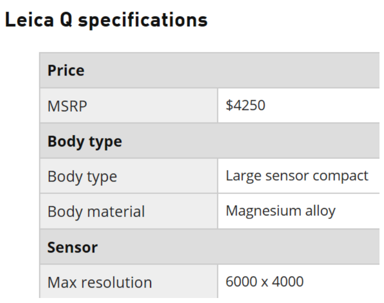
Table 1: Quality evaluation of synonyms. Recall is defined as the avg. number of synonyms produced per entity.

3 Web Table Services

We describe the key techniques that enable the web table search and web table answer services. Both services rely on the *web table extraction and understanding* component which we describe first. The web table search service takes a keyword query as input and returns a *ranked list of web tables* relevant to the query. Since the tables are already extracted, the key remaining challenge is to provide high quality ranking over those tables. Ranking features used in web page ranking or prior work on web table search are not adequate for our purpose [19]. We develop novel ranking features to address this challenge. We describe them in the second subsection. The web table answer service also takes a keyword query (a general web search query) but returns *a single table* if it is the best possible answer (among all possible types of answers as well as top algorithmic search result), otherwise it returns nothing. The main challenge here is to provide table answers with a high precision and good coverage. We develop a novel approaches to address this challenge which we describe in the last subsection.



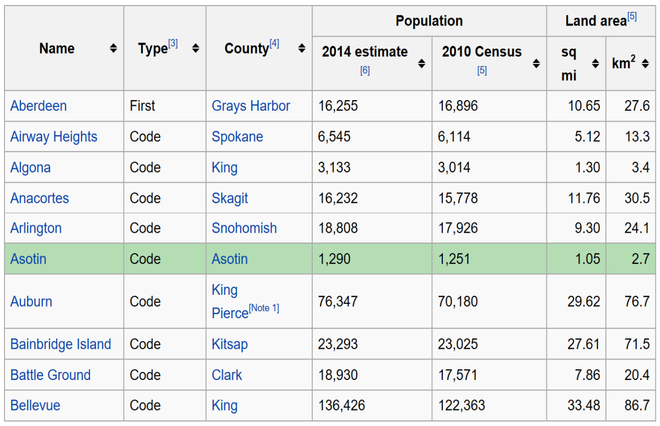
(a)



Leica Q specifications

Price	
MSRP	$4250
Body type	
Body type	Large sensor compact
Body material	Magnesium alloy
Sensor	
Max resolution	6000 x 4000

(b)



Name	Type[3]	County[4]	Population		Land area[5]	
			2014 estimate [6]	2010 Census [5]	sq mi	km^2
Aberdeen	First	Grays Harbor	16,255	16,896	10.65	27.6
Airway Heights	Code	Spokane	6,545	6,114	5.12	13.3
Algona	Code	King	3,133	3,014	1.30	3.4
Anacortes	Code	Skagit	16,232	15,778	11.76	30.5
Arlington	Code	Snohomish	18,808	17,926	9.30	24.1
Asotin	Code	Asotin	1,290	1,251	1.05	2.7
Auburn	Code	King Pierce[Note 1]	76,347	70,180	29.62	76.7
Bainbridge Island	Code	Kitsap	23,293	23,025	27.61	71.5
Battle Ground	Code	Clark	18,930	17,571	7.86	20.4
Bellevue	Code	King	136,426	122,363	33.48	86.7

(c)

Figure 7: Different types of HTML tables. (a) is a layout table, (b) is an attribute-value table, (c) is a relational table.

3.1 Prior Work on Web Tables

Web table extraction: There is significant work on web table extraction in the literature [6, 5, 21, 20, 14, 11]. One of the key challenges is to distinguish between the different types of tables, viz. layout, relational and attribute-value tables. Most of the above works formulate the problem as a machine learning classification task and design features for the task. Examples of features that distinguish relational tables from other ones are cell value consistency along columns and the number of empty cells [6, 5, 21]. We adopt a similar approach in our paper. While there is overlap between our features and those proposed in earlier work, we also employ novel features like header features to improve accuracy.
Web table search: Venetis et. al. developed a seminal system for keyword search on web tables [19]. They annotate each table with column labels that describe the class of entities that appear in that column (using a isA database). Then they look for matches between the query keywords and the annotations as well as the column names. This results in much better quality than simply piggybacking on Google's ranking of web pages. Their approach is not adequate for our services for multiple reasons. For the web table search service, some classes of queries require us to find matches inside the data cells (e.g., row-subset queries, entity-attribute queries). The above work does not consider such matches. Second, for the web table answer service, there is a hard requirement of 98% precision while maintaining good coverage. The above approach is not optimized for this requirement.

3.2 Web Table Extraction and Understanding Component

The goal of this component is to extract tables from Bing's web crawl corpus and perform table understanding and annotation. These tables are subsequently used in web table search and web table answer services. We first obtain the raw HTML tables (i.e., elements enclosed by the <table></table> tags) by parsing each document in the crawl corpus into a DOM tree. The main challenge is that not all the raw HTML tables contain valuable data; a majority of them are used for layout purposes (e.g., page layout, form layout). Figure 7(a) shows an example of a layout table. We need to identify and discard these tables. Among the valuable tables, there are two main types of tables, namely *relational* and *attribute-value* tables. A relational table is one where each row corresponds to a named entity and the columns correspond to different attributes of the entities [5, 6]. Figure 7(c) shows an example of a relational table where each row corresponds to a city (in Washington) and the columns correspond to different attributes like the county the city belongs to, the city's population and the city's land area. On the other hand, an attribute-value table contains different attributes and its values of a single entity [11]. Figure 7(b) shows an attribute-value table for the entity 'Leica Q'. We need to distinguish between the two types of tables in order to understand their semantics which in turn is necessary to provide high quality table search and table answer services. These are challenging problems as they cannot be accomplished via simple rules [5].

We first present our approach to distinguish between the different types of tables. Like prior work, we formulate the problem as a machine learning classification task and propose features for the task [6, 5, 21, 20]. We improve upon those works by proposing novel features like header features. We describe all the features below for the sake of completeness.

Distinguishing among different types of tables We identify several features to distinguish the different types of tables. We categorize the features as follows:

• *Column-wise homogeneity*: One of the main features that distinguishes relational tables from attribute-value tables is column-wise homogeneity. The values in a column of a relational table are typically homogeneous as it contains values of different entities *on the same attribute*. For a string-valued attribute, all cells have string values and their string lengths are similar to each other (e.g., three leftmost columns in table in Figure 7(c)). Similarly, for a numeric attribute, all cells have numeric values (e.g., four rightmost columns in table in Figure 7(c)). A column in a relational table typically does not contain a mix of string and numeric values. On the other hand, the second (counting from left) column of an attribute-value table typically contains a mix of string and numeric values as it contains values on *different attributes*. For example, the second column in the table in Figure 7(b) contains such a mix. We capture the above observations by proposing the following features: (i) fraction of string-valued cells in first and second columns (ii) fraction of numeric-valued cells in first and second columns (iii) mean and standard deviation of string lengths of cells in first and second columns (iv) mean and standard deviation of string lengths of cells averaged across all columns. A high value on either (i) or (ii) represents homogeneity; we capture that by computing a derived feature that computes the max between (i) and (ii).

• *Row-wise homogeneity*: Layout tables are also often column-wise homogeneous, so column-wise homogeneity alone cannot distinguish between relational and layout tables. We use row-wise homogeneity for that purpose. All rows in a relational table typically have the same number of cells and they are typically non-empty. On the other hand, rows in a layout table might have different number of non-empty cells. For example, in the layout table in Figure 7(a), the first, second and third rows have 2, 3 and 1 non-empty cells respectively. We compute the standard deviation of the number of cells (and non-empty cells) in different rows as features.

• *Row and column counts*: Row and column counts also help distinguish between the different types of tables. Layout tables often have a very small number of rows and/or a very small number of columns (e.g., the layout table in Figure 7(a) in Figure has only 3 rows). Attribute-value tables typically have 2 columns and a small number of rows. On the other hand, relational table often have much larger number of rows and more than 2 columns. We compute number of number of rows and number of columns as features.

• *Presence of header*: If a table contains a header row that is explicitly marked by `<th>` or `<thead>` tags and it "aligns" with the rest of the table (i.e., has the same number of columns as other rows), it is most likely a relational table. This is especially true if there is a column where all the cell values are numeric except the cell value in the `<th>` or `<thead>` row (which has a string value). We compute boolean features to capture the above insights.

We manually labeled about 5000 tables as relational, layout and attribute-value and trained a decision forest model on those labeled examples. We tuned our model for high precision. Our relation classifier has a precision of 96%. We focus on relational tables in our project as we believe that these tables are most useful for joining with business data tables in spreadsheet scenarios; we plan to include attribute-value tables in the future. The rest of discussion focuses on relational tables.

This component also tries to "understand" the relational tables. Each relational table typically has one column that contains the names of the entities that correspond to the rows of the table [19, 20]. We refer to it as the *entity column* of the table. For example, in the table in Figure 7(c)), the leftmost column is the entity column. It is important to pinpoint the entity column for high-quality search. Consider two entity-attribute

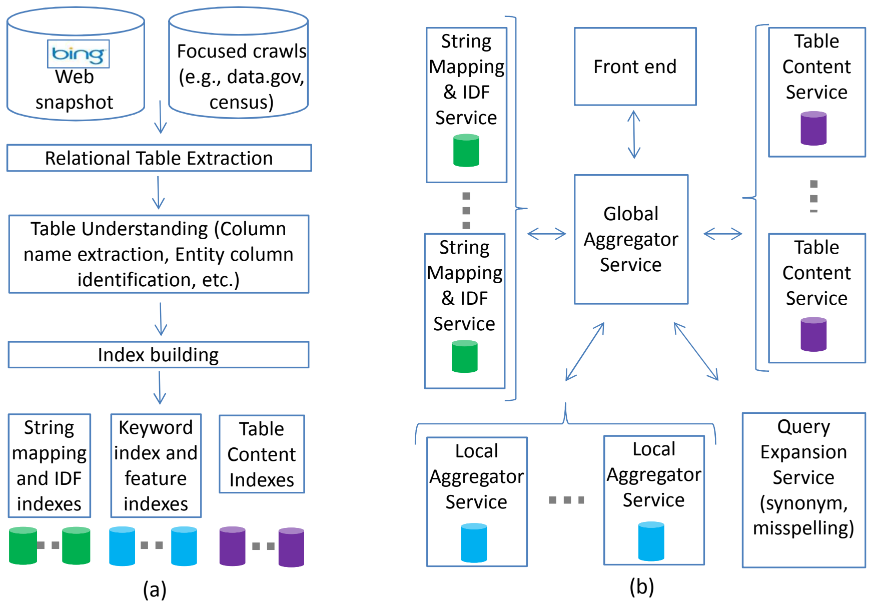

bing
Web snapshot
Focused crawls (e.g., data.gov, census)
Relational Table Extraction
Table Understanding (Column name extraction, Entity column identification, etc.)
Index building
String mapping and IDF indexes
Keyword index and feature indexes
Table Content Indexes
(a)
String Mapping & IDF Service
String Mapping & IDF Service
Front end
Global Aggregator Service
Table Content Service
Table Content Service
Local Aggregator Service
Local Aggregator Service
Query Expansion Service (synonym, misspelling)
(b)

Figure 8: System architectures for (a) Table extraction and understanding and (b) Table search service.

queries: {`aberdeen population 2014`} and {`grays harbor population 2014`}. Both queries "match" the first data row in the table. However, it is a correct match for the first query (as it contains the 2014 population of Aberdeen) but not for the second query (as it does not contain the 2014 population of Grays Harbor). This can be ascertained only if we know the entity column. We identify the entity column using machine learning techniques. Another example of table understanding is column name extraction, especially when the column names are not marked explicitly via `<th>` or `<thead>` tags. Figure 8(a) shows the various subcomponents of this component.

3.3 Web Table Search Service

The web table search service takes a keyword query as input and returns a ranked list of web tables relevant to the query. The architecture as well as the query processing of the web table search service are similar to that of a general web search engine. The architecture is shown in Figure 8(b). The indexed items are web tables instead of web pages. The inverted index over web tables as well indexes containing features about web tables (e.g., number of rows, PageRank) are distributed over many computers, each corresponding to a subset of the web table corpus. Each of those computers runs a local aggregator service that traverses the inverted index and performs top-k ranking among the table subset in that computer. When a query arrives, the global aggregator service distributes the query to the local aggregator services to search simultaneously. It then consolidates all the local top-k results and computes the global top-k. Finally, it invokes the table content services (which stores the full content of each table) to generate query-dependent snippets for the top-k tables.

The key challenge here is to provide a ranking of high quality. Ranking features used in web page ranking or prior work on web table search are not adequate for our purpose [19]. The above features are adequate when the user searches based on the description of the tables; here, we only need to look for keyword matches in the description fields (e.g., page title, table caption) and column names. However, there are some classes of queries which require us to find matches inside the data cells as well as the *alignment* of those matches (e.g., in the same column). One such class is *row-subset queries* where the ideal answer is a *subset of rows in a bigger table* and that subset can be obtained by filtering on the value of an attribute. Consider the query 'largest software companies in usa'. The table in Figure 2(b) contains the largest software companies from all over the world (including several from USA) and is hence relevant. However, 'usa' is not mentioned anywhere in the page except in the cells of the ‘Headquarters’ column. To identify the above table as a relevant one, we need to compute a table alignment feature that indicates ’usa’ occurs multiple times in different rows of the same (non-entity) column of the table. Prior works do not compute such features and hence fail to return such a table. We incorporate such features in our table search engine.

Another such class is *entity-attribute queries*. Consider a query ‘aberdeen population’ where the user is looking for a table containing population of Aberdeen. The ideal table should contain ‘aberdeen’ in a cell in the entity column and ‘population’ as a column name of a non-entity column. Once again, the above approaches fail to return the relevant table. We incorporate such table alignment features in our service. To compute these features, we need to know the row and column indexes of keyword hits in data cells and column indexes of keyword hits in column names. A document search engine does not store such fine grained information; we design our inverted index to store such information.

3.4 Web Table Answer Service

The web table answer service takes a keyword query (a general web search query) and returns a single table if it the best possible answer (among all possible types of answers as well as top algorithmic search result). Otherwise, it does not return any answer. The main challenge is to have high precision and still having significant coverage.

One of the main challenges is that a perfect match of the query with a table does not guarantee that the table is the best possible answer. Table answer may not be the best type of answer at all; the top algorithmic search result or some other type of answer (e.g., an entity information card or a passage answer) might be better. Consider the query ‘washington state climate’. The table with caption “Climate data for Washington State (1895-2015)” in `en.wikipedia.org/wiki/Climate_of_Washington` is a perfect match as Bing returns that page at second position and both the table caption and page title perfectly matches with the query. But a passage answer is a better and more compact answer; both Bing and Google returns a passage answer for this query. We address this challenge by following a two-step approach. In the first step, we determine whether table answer is the *best type* of answer; this is irrespective of whether such a table answer exists or not. If yes, we try to find such a table in the second step. We briefly describe the two steps.

Determine whether table answer is best answer type: We identify several classes of queries for which table answer is the best type of answer. One such class is *list-intent queries*. Such a query seeks for two or more entities. An example is ‘largest software companies in usa’ as shown in Figure 2(b). Typically a table containing all the desired entities and their important attributes is the best possible answer. Such queries contain the desired entity type in plural; we obtain the list of possible desired entity types from Bing’s knowledge graph. For example, the desired entity type in the above query is ‘company’. However, this is not a sufficient condition. For example, ‘us companies outsourcing’ is not a list-intent query although it contains an entity type in plural. We observe that for list-intent queries, the rest of the query, specifically the pre-modifier (the part preceding the desired entity type) and post-modifier (the part following it), either specifies constraints on the entities (to narrow down the desired entities) or a ranking criterion. For example, ’software’ and ’in usa’ are constraints and ’largest’ in the ranking criterion. The constraint can be either an adjective or an entity that is “compatible” with the desired entity type. For example, ‘software’ is an adjective which is compatible with company while ’usa’ is a entity compatible with company. ’Usa’ is compatible with company as it is a location entity and companies are often constrained by location. On the other hand, a film or album entity will not be compatible with company as companies are typically not constrained by films or albums. The challenge is to check whether the pre-modifier and post-modifier satisfy the above criteria. We address this challenge in two steps. We first create a dictionary of all adjectives compatible with any entity type; we obtain this from Probase [22]. We also create a dictionary of all entities compatible with any entity type; we first manually curate constraining entity types that are compatible with any query entity type (e.g., location entity type is compatible with company) and then obtain the entities of the former types from Bing’s knowledge graph. We also curate a list of ranking keywords. In the first step, we identify matches of compatible adjectives, compatible entities and ranking keywords in the premodifier and postmodifier. Subsequently, we use a probabilistic graphical model to discover the holistic set of matches that best "covers" the query. If the coverage exceed a certain threshold, we classify the query as list-intent query.

Another class of queries for which table answer is the best type of answer is *superlative queries* (e.g., 'largest software company in usa'). This class is closely related to the class of list-intent queries; we use similar techniques to identify these queries. Under certain conditions, table answer is the best type of answer for *entity-attribute queries* as well; we identify this class of queries as well. The output of this step is a decision whether the query belongs to one of the above classes and, if yes, the parsed query.

Finding table answer: Given a parsed query belonging to one of the above classes, this step tries to find a table answer. We first obtain a set of candidate tables among which we try to find the table answer: these are the tables extracted from the top k ($k \leq 10$) pages returned by Bing for the query. Finding a table answer within the candidate set is challenging. One challenge is that there are multiple tables within the page, and often many of them are similar from the perspective of keyword match. Consider the query "tom cruise movies". There are two tables from `www.movies.com/actors/tom-cruise/tom-cruise-movies/p284917` in the candidate set: one containing the movies that Tom Cruise acted in and the other containing the actors that Tom Cruise has worked with. Both tables have hits for all the keywords. But only the first table is the right answer since the desired entity type is movie. We address this challenge by leveraging the fine-grained understanding of roles of keywords in query such as desired entity type, constraining entity, constraining concept and ranking keyword. In this example, the desired entity type matches with the column name of the entity column of the first table but there is no match with the entity column of the second one. We enrich our ranking features with above roles to make such distinctions.

Another challenge is that the table answer not only needs to be the best table among the candidate tables but also better than the top algorithm result in Bing. For example, for college ranking queries, the top Bing answer is usually the right page from US News. However, sometimes the ranking list in the US News page is not formatted as a table or the table extractor fails to extract the table from the page. We might be able to find a table among the other candidate tables that perfectly matches user's intent but it is from a less well-known source. We do not want to show such a table answer above the proven top result.. We identify such cases by using the click features (click count, click through rate) of both the page containing the candidate table and the most clicked Bing search result for that query.

4 Conclusion

In this paper, we provide a brief overview of the data services we have been developing by leveraging Bing's data assets. Specifically, we described two data services: synonym service and web table service. Both services have been successfully integrated into various Microsoft products and services. One of the key learnings is that commercial applications require data services with very high precision (high nineties), otherwise it would result in user dissatisfaction. While it is usually straightforward to obtain about 80% precision (for both synonym and web table services), it is hard to obtain 95+% precision (while still having good coverage). We need novel approaches to accomplish that level of precision as we described in the paper.

Acknowledgements

We thank Tao Cheng, Dong Xin and Venkatesh Ganti for their core research contributions to the Synonym project. We thank David Simpson, Guihong Cao, Han Wang, Yi Li and Jiang Wu from Bing who collaborated closely with us to ship the web table answers in Bing. We thank Minghui (Jason) Xia from Bing whose team integrated our synonyms data into Bing's entity linking technology. We thank James Finnigan, Konstantin Zoryn, Jean-Sebastien Brunner and Dennis Churin from SQL Server Information Services team who shipped web table search service in Excel PowerQuery. Finally, we thank Arnd Christian Konig and Vivek Narasayya for their feedback on the initial draft of the paper.

References

[1] Bing Synonyms API. `http://datamarket.azure.com/dataset/bing/synonyms`, 2012.

[2] Search public data (Power Query) - Excel. `https://support.office.com/en-us/article/Search-public-data-Power-Query-12e0d27e-2e91-4471-b422-c20e75c008cd?ui=en-US&rs=en-US&ad=US`, 2013.

[3] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Proceedings of the 2004 international conference on Current Trends in Database Technology*, 2004.

[4] Ricardo Baeza-Yates and Alessandro Tiberi. Extracting semantic relations from query logs. In *Proceedings of KDD*, 2007.

[5] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[6] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. Uncovering the relational web. *WebDB*, 2008.

[7] Kaushik Chakrabarti, Surajit Chaudhuri, Tao Cheng, and Dong Xin. A framework for robust discovery of entity synonyms. In *SIGKDD*, 2012.

[8] Chia-Hui Chang, Mohammed Kayed, Moheb Ramzy Girgis, and Khaled Shaalan. A survey of web information extraction systems. *Transactions on Knowledge and Data Engineering*, 2006.

[9] Surajit Chaudhuri, Manoj Syamala, Tao Cheng, Vivek Narasayya, and Kaushik Chakrabarti. Data services for e-tailers leveraging web search engine assets. In *ICDE*, 2013.

[10] Tao Cheng, Hady W. Lauw, and Stelios Paparizos. Entity synonyms for structured web search. *Transactions on Knowledge and Data Engineering*, 2011.

[11] Eric Crestan and Patrick Pantel. Web-scale knowledge extraction from semi-structured tables. In *WWW*, 2010.

[12] Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. Probabilistic query expansion using query logs. In *Proceedings of WWW*, 2002.

[13] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 2001.

[14] Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krüpl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *WWW*, 2007.

[15] Yeye He, Kaushik Chakrabarti, Tao Cheng, and Tomasz Tylenda. Automatic discovery of attribute synonyms using query logs and table corpora. In *Proceedings of WWW*, 2016.

[16] Jianhua Lin. Divergence measures based on the shannon entropy. *Transactions on Information Theory*, 1991.

[17] Sunita Sarawagi. Information extraction. *Foundations and Trends in Databases*, 2008.

[18] Bilyana Taneva, Tao Cheng, Kaushik Chakrabarti, and Yeye He. Mining acronym expansions and their meanings using query click log. In *WWW*, 2013.

[19] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *PVLDB*, 4(9):528–538, 2011.

[20] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu. Understanding tables on the web. In *ER*, 2012.

[21] Yalin Wang and Jianying Hu. A machine learning based approach for table detection on the web. In *WWW*, 2002.

[22] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: A probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.